An adaptive and dynamical neural network for machine remaining useful life prediction

Abstract—Recently, many neural networks have been proposed for machine remaining useful life (RUL) prediction. However, most network architectures of existing approaches are fixed. Since the sequential information depends on the input data and distributes differently, these fixed networks which cannot be dynamically adjusted according to the input data, may not be able to capture these sequential information well, resulting sub-optimal performances. To mitigate this issue, we propose an adaptive and dynamical neural network (AdaNet), which can dynamically adjust its architecture according to the input data. Neural network is generally determined by kernel size, depth and channel size. In this paper, we aim to enable our proposed AdaNet to adjust its kernel size and channel size dynamically. Firstly, we explore to adapt the deformable convolution to time series data, which allows the convolutional kernel to change according to the feature map. With this deformable convolution, the convolutional kernels in AdaNet become adjustable, which is beneficial to fully exploit the sequential information in time series data, leading to accurate RUL prediction. Additionally, a channel selection module is devised, which can selectively activate the feature channel according to the input, further improving the performance of our AdaNet. Extensive experiments have been carried out on the C-MAPSS dataset, demonstrating that our proposed AdaNet achieves state-of-the-art performances.

Index Terms—Remaining useful life prediction, deep learning, neural network, time series

I. INTRODUCTION

P ROGNOSTIC health management (PHM) becomes more and more attractive in the industry 4.0, which refers to a series of maintenance policies to ensure that an industrial system works in a normal condition. In PHM, the machine remaining useful life (RUL) prediction is an essential task, which is used to predict the remaining useful life of an industrial component or a machine, preventing the industrial system from system failures and providing a reliable environment [1]–[12]. In recent decades, many methods have been proposed to predict the RUL, which can be roughly divided into two categories, model based methods [11], [12] and data driven methods [1]–[10].

Model based methods [11], [12] aim to utilize prior knowledge to build models for industrial systems, which requires expert knowledge for a specific system or an industrial component. However, with industrial systems becoming complex, it is infeasible to apply these methods to real industry. In comparison, without the needing of prior knowledge, data driven methods [1]–[10] are able to model these industrial systems by collecting enough sensor data. Benefited from the advance of the deep learning technology [13]–[15], data driven methods recently show impressive performances in the RUL prediction task and achieve state-of-the-art results on some classic benchmarks [16], [17].

Although existing data driven methods achieve promising progresses on the RUL prediction task, the architecture of most approaches is fixed and cannot be dynamically adjusted according to the input data. In the RUL prediction task, it is essential to exploit the temporal dependencies from the time series data and capture the sequential information. However, the temporal dependencies are relevant to the time series data and may distribute differently. Since the fixed neural network often focuses on capturing those temporal dependencies which often appear in the training set, it is difficult to apply these networks to learn arbitrary type of temporal dependencies, especially for the rare but useful sequential information appearing during the training process. Furthermore, current neural networks are trained on a training data set and predict the RUL on a test data set. Since there is a domain gap between these two data sets, these fixed networks may not be able to well adapt to the test data set. This further degenerates their performances.

To mitigate these issues above, we propose an adaptive and dynamical neural network called AdaNet for the RUL prediction task. Different from existing methods, our proposed network is able to adjust its architecture according to the input data and effectively capture the sequential information, resulting accurate RUL prediction. With the strong capacity of feature representation, convolutional neural network (CNN) has been widely used in many applications [15], [18]. We also propose our AdaNet based on CNN. CNN is usually determined by kernel size, depth and channel size. In this paper, to allow the CNN to adjust itself, we propose to adapt two kinds of layers: deformable convolution layer and channel selection module, which allow our AdaNet to adjust its kernel size and channel size according to the input data.

In our AdaNet, to allow the convolutional kernel to be changeable, we explore the deformable convolution (DC) [19] on the time series data, which is originally proposed for the geometric transformation in computer vision. To present the DC layer clearly, we compare the DC layer and the normal convolution layer in Fig. 1. A normal 2D 3×3 convolution is shown in Fig. 1 (a) and two cases of 2D 3×3 DC layer are displayed in Fig. 1 (b) and (c). Compared with the normal convolution, the DC layer is able to recognize the distribution of the useful features and move its kernel sampling location to corresponding positions. Fig. 1 (b) indicates a case of DC layer which increases its receptive field and Fig. 1 (c) represents a
DC layer in a general case. Based on the original DC layer, instead of modeling the geometric variation in images, we alternatively adopt it to time series data and enable this DC layer to dynamically capture the sequential information. To the best of our knowledge, we are the first to develop the DC layer to the RUL prediction task. Benefited to the DC layer, our AdaNet can effectively capture the sequential information and predict the RUL accurately.

Moreover, a channel selection (CS) module is proposed in our AdaNet, which further improves the flexibility of a neural network. CNN is proposed to hierarchically model objects, where features from different channels are used to represent different semantic information [20]. To accurately represent objects, these produced features sometimes may be redundant. For a specific target, it may not be necessary to produce all these features and some extra features sometimes even affect the performance of a CNN. To address this issue, we devise a channel selection module which can selectively activate the feature channel according to the input data. Our CS module is developed according to the attention scheme [21]. Given by input data, the useful features are activated, while other redundant features are suppressed. With our proposed CS module, the performances of our AdaNet can be further improved. Extensive experiments are carried out on the C-MAPSS dataset [16] to verify the effectiveness of our proposed AdaNet. Through experiments, our AdaNet shows the state-of-the-art performances.

In summary, our contribution can be summaries as follows:

1) Different from the fixed networks in existing methods, an AdaNet is proposed, where we explore the deformable convolution layer in the RUL prediction task. With the deformable convolution layer, our AdaNet is able to dynamically move the sampling location of the convolutional kernel and well capture the sequential information, achieving satisfactory performances for RUL prediction.

2) To further improve the flexibility of our AdaNet, we devise a channel selection module, which is able to selectively activate the feature channel and improve the quality of the produced feature map.

3) To verify the effectiveness of our proposed AdaNet, we conduct massive experiments on the C-MAPSS benchmark and obtain state-of-the-art performances.

The remaining part of this paper is organized as following. Literature review on relevant methods is discussed in Section II. Our proposed AdaNet is presented in Section III. The experimental results and related settings are discussed in Section IV. Finally, conclusion and future work are provided in Section V.

II. LITERATURE REVIEW

Benefited from the development of the deep learning technology, data driven approaches [1]–[10] become more and more attractive for the RUL prediction. These approaches fall into three categories: CNN based methods, Recurrent Neural network (RNN) based methods and hybrid methods.

CNN based methods usually propose to utilize CNNs to extract feature representation for the input data and predict the RUL. Li et al. [7] proposes a CNN based method, where a time window technology and data normalization are applied for better performances. Yang et al. [5] devises a double-CNN based framework, which divides the RUL prediction into two stages: incipient failure point identification and RUL prediction. After that, Song et al. [3] utilizes the attention mechanism to improve the quality of data and suppresses the data noise. KDnet [2] applies the knowledge distillation to distill the knowledge from a RNN to a CNN, and shows impressive RUL prediction accuracy. Jin et al. [9] devises a Bi-LSTM based two-stream neural network, where the handcrafted feature flow is proposed and fused with the raw data to predict the RUL. In DBNet [1], a branched network is proposed, which jointly learns to classify the feature mode and predict the RUL. Ren et al. [10] develops a multi-head network, where the time constrain between different time internals are considered. Benefited from the strong feature representation, CNN based approaches show impressive performances. However, the sequential information cannot be fully exploited due to the parallel scheme of CNN. To alleviate this issue, some RNN based methods are proposed.

Compared with CNN based methods, RNN based methods target to use the RNN to capture the temporal dependencies from the time series data. There are variants of RNN, like long short-term memory (LSTM) and gated recurrent unit (GRU). Deep LSTM [8] and Liao et al. [22] explore the LSTM on the RUL prediction. Based on these approaches, Zhang et al. [23] and Huang et al. [24] improve the LSTM based methods and apply bi-directional LSTM to the RUL prediction. However, since RNN processes data recursively, the training and testing of RNN based methods is time consuming. This also limits the depth of the RNN based approaches.

To utilize the advantages of RNN and CNN based methods and alleviate the drawbacks of these approaches, some hybrid methods which combine RNN and CNN together, are proposed to achieve better performances. HDNN [25] devises to combine the features from a CNN and a LSTM together to predict the RUL. In Liu et al. [6], the LSTM is used as an encoder and a CNN is adopted as a feature decoder to predict the RUL. TCMN [26] firstly uses temporal convolutional layers to capture temporal features and then forward to several LSTM for feature enhancement. AGCNN [4] proposes to use a bi-directional GRU as a feature encoder and forwards the produced features to the CNN to predict the RUL.
Recently, several approaches [27], [28] propose to use some auxiliary tasks to improve the RUL accuracy. DGRU [27] applies the conditional generative adversarial network for data augmentation and proposes a network consisting of several GRUs to predict the RUL. Jang et al. [28] uses the self-supervised learning technology to improve its performances.

Although many deep learning based approaches are proposed to predict the RUL, the networks of these methods are fixed and cannot adjust their architectures according to the input data. To address this issue, we propose an AdaNet, which is able to dynamically adjust the architecture and well capture the sequential information, predicting the RUL accurately.

III. METHODOLOGY

In this section, we present our proposed AdaNet for the RUL prediction task. Given by the time series data $X \in \mathbb{R}^{T \times S}$, the RUL prediction task can be modeled as following:

$$Y = f(X, \theta),$$

where $f$ denotes the neural network, $\theta$ represents the parameters of the network and $Y$ is the target. Different from existing approaches which devise fixed neural networks, we aim to propose a neural network to accurately predict the RUL via adjusting the network architecture online.

The pipeline of our AdaNet is illustrated in Fig. 2. Our AdaNet uses several DC layers and CS modules to adjust the network architecture online. The DC layer allows our AdaNet to dynamically adjust its convolutional kernel, while the CS module enables our AdaNet to change the feature channel size online.

A. Deformable Convolution

In general, given by a location, a normal convolutional layer processes the convolution operation on a fixed sampling region. Since sequential information may distribute differently, this fixed sampling location may not be able to include all necessary information and thus limit the quality of the encoded features. To solve this issue, we adopt the DC layer which dynamically adjusts the sampling location, to learn the sequential information from time series data.

For a pre-defined sampling region $\Omega$ and a location $\mathbf{p}$ in the input data $X$, a normal 2D convolution process can be formulated as:

$$Y(\mathbf{p}) = \sum_{\mathbf{p}_i \in \Omega} W(i) \cdot X(\mathbf{p} + \mathbf{p}_i),$$

where $\mathbf{p}_i$ denotes the relative sampling location, like (-1, 0) and (0, 1), $i$ represents the sampling index, and $W$ indicates the learnable weights of a convolutional kernel.

Compared with the normal convolution process, DC layer [19] provides offsets for the relative sampling location $\mathbf{p}_i$ and allows the sampling location to move. Given by a group of sampling offsets $\{\Delta \mathbf{p}_i | i = 1, 2, \ldots N\}$, where $N$ denotes the number of sampling location, a 2D DC layer is defined as:

$$Y(\mathbf{p}) = \sum_{\mathbf{p}_i \in \Omega} W(i) \cdot X(\mathbf{p} + \mathbf{p}_i + \Delta \mathbf{p}_i).$$

To effectively implement the DC layer, we divide the DC layer into two parts: offset prediction and convolution adjustment, which is shown in Fig. 2. The offset predict is used to predict the offsets for each sampling location for the convolution kernel. The convolution adjustment is utilized to adjust the moved kernel to the input data.

Offsets prediction. To predict the offsets $\Delta \mathbf{p}_i$, a shallow network is adopted. According to Eq. 3, the predicted offset $\Delta \mathbf{p}_i$ is related to two factors: the location $\mathbf{p}$ in the input data and the relative sampling location $\mathbf{p}_i$. This indicates that the offsets may be different for different $\mathbf{p}$ and $\mathbf{p}_i$. In a 2D DC layer, a predicted offset $\Delta \mathbf{p}_i$ includes two moving offsets: $\Delta \mathbf{p}_{ix}$ and $\Delta \mathbf{p}_{iy}$. To accurately predict the offsets, we propose a shallow network to predict $2 \times |\Omega|$ offsets for each location $\mathbf{p}$. Additionally, the architecture of this shallow network may affect the accuracy of the predicted offsets, especially for the value range of the predicted offset. In this paper, for
This computation process can be defined as follows:

\[ X(p) = \sum_{p_i \in \Omega} W(i) \cdot \hat{X}(p + p_i + \Delta p_i), \quad (4) \]

where \( \hat{X} \) is estimated via the bilinear interpolation as below:

\[ \hat{X}(p') = \sum_{q' \in \Phi} f(q'_x, p'_x) \cdot f(q'_y, p'_y) \cdot X(q'), \quad (5) \]

where \( p' = p + p_i + \Delta p_i \), \( \Phi \) denotes the set of all integral locations around \( p' \), and \( f(x, y) = \max(0, 1 - |x - y|) \). For example, for a moved sampling position \((3.5, 4.7)\) in a 2D space, the four adjacent and integral locations in \( \Phi \) should be \((3.0, 4.0), (4.0, 4.0), (3.0, 5.0)\) and \((4.0, 5.0)\).

For a 2D space, since there are only four integral locations around a fractional spatial location, the computation process is efficient.

**B. Channel Selection Module**

Different feature channels may encode different information. For a series of specific input data, parts of feature channels may be redundant and affect the feature quality. To solve this problem, a channel selection layer is presented according to the attention scheme [21] in this subsection.

As shown in Fig. 2, our proposed channel selection module consists of two parts: channel embedding and channel weight. To present our channel selection module, the detailed structure for this module is illustrated in Fig. 3. To selectively activate the feature channel clearly, the input feature is firstly forwarded to an average pooling layer. After that, our proposed channel embedding module is applied on the pooled feature vector, which is followed by a ReLU layer. Then, this embedded feature is passed to a channel weight module and a Sigmoid layer to produce the weights for each feature channel. Finally, the produced weight by this CS module is multiplied to the original feature map to selectively activate the feature channels. The process of our CS module can be formulated as:

\[
X'_o = X_o \sigma(W_{cw} \delta(W_{ce} \bar{X}_o)), \quad (6)
\]

where \( \bar{X}_o \) is computed via a global pooling layer, which is defined as:

\[
\bar{X}_o = \frac{1}{H \times W} \sum_{i=1}^{H} \sum_{j=1}^{W} X_o(i, j). \quad (7)
\]

In Eq. 7, \( H \) and \( W \) denote the spatial size of a produced feature map. \( W_{ce} \) indicates the learnable weights for the channel embedding module, while \( W_{cw} \) represents the weights for the channel weight module. \( \delta \) and \( \sigma \) denote a ReLU layer and a Sigmoid layer, respectively.

For different spatial location, the importance among feature channels may be different. For computation efficiency, we hypothesize that the importance among feature channels is invariant to the spatial location. A global average pooling operation is applied on the input feature map \( X_o \in \mathbb{R}^{H \times W \times C} \) to produce a feature vector \( \bar{X}_o \in \mathbb{R}^C \) which is used as a feature summary. The feature embedding module is used to encode the relationship between feature channels via multiplying its weight \( W_{ce} \in \mathbb{R}^{r \times C \times C} \) with \( \bar{X}_o \). After a ReLU layer, the produced feature vector is passed to the channel weight module with learnable weight \( W_{cw} \in \mathbb{R}^{C \times r} \). The \( r \) in \( W_{ce} \) and \( W_{cw} \) denotes a feature ratio, which is used for feature dimensionality reduction. The channel weight is then produced by a non-linear layer, Sigmoid. The weights produced from the Sigmoid layer are used to improve the original features, where useful feature channels are amplified, while redundant feature channels are suppressed.

**C. Neural Network Architecture**

In this subsection, we present the detailed structure of our AdaNet. Our proposed AdaNet is composed of DC layers, CS modules, normal convolutional layers and fully connected layers. The detailed structure is listed in Table I, where the hyper-parameters for a normal convolution or a DC layer are denoted as (kernel size, channel number, stride value, padding value).

As listed in Table I, our AdaNet consists of six layer groups, two fully connected layers and a regressor. These layer groups are used as a feature encoder, where the structure of a layer group is visualized in Fig. 4. For computation efficiency, we apply the DC layer to some layer groups, while use the normal convolutional layer in others.
In each layer group, the input is firstly forwarded to a DC or normal convolutional layer followed by a Batch Normalization (BN) layer. To reduce the computation cost, the stride value for layer group-1 is set as 2. Following the protocol of classic networks [14], [15], [29], the original feature channel size increases from 16 to 256 gradually with the network depth increasing. The channel size listed in Table I is the maximum feature channel size, and the actual channel size will be adjusted dynamically by our CS module. The stride value for layer group-3_1 and layer group-4 are set as 2 for further decreasing the size of the feature map. After these layer groups, two fully connected layers serve as a feature decoder and a regressor is used to predict the RUL.

In each layer group, the input is firstly forwarded to a DC or normal convolutional layer followed by a Batch Normalization (BN) layer. To reduce the computation cost, the stride value for layer group-1 is set as 2. Following the protocol of classic networks [14], [15], [29], the original feature channel size increases from 16 to 256 gradually with the network depth increasing. The channel size listed in Table I is the maximum feature channel size, and the actual channel size will be adjusted dynamically by our CS module. The stride value for layer group-3_1 and layer group-4 are set as 2 for further decreasing the size of the feature map. After these layer groups, two fully connected layers serve as a feature decoder and a regressor is used to predict the RUL.

### IV. Experiments

To verify the effectiveness of our proposed AdaNet, we conduct extensive experiments and discuss the related experimental results in this section.

#### A. Dataset and Experimental Settings

1) Dataset: A well-known benchmark, C-MAPSS [16] is used to verify the performance of our AdaNet. C-MAPSS benchmark records the run-to-failure degradation process of the turbofan engine, which is devised by NASA. In C-MAPSS, these time series data comes from 21 sensors installed on the turbofan engine, like temperature, pressure and speed. The diagram of the turbofan engine is illustrated in Fig. 5.

![Fig. 5. Illustration for the turbofan engine in the C-MAPSS benchmark](image-url)

There are four subsets, FD001, FD002 FD003 and FD004, in this benchmark, where the detailed information is listed in Table II. Each subset is further divided into a training set and a testing set, where hundreds of trajectories for the engine degradation process are involved in them. Among these four subsets, FD001 is the simplest subset, where only one type of condition and one type of fault mode are involved. The number of training trajectory and testing trajectory are 100. In comparison, FD004 contains much more complex scenarios, where six different conditions and 2 fault modes are included.

![Diagram of the turbofan engine](image-url)

#### TABLE I

<table>
<thead>
<tr>
<th>Layer Name</th>
<th>Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>Layer Group-1</td>
<td>(6, 16, 2, 1)</td>
</tr>
<tr>
<td>Layer Group-2</td>
<td>(6, 64, 1, 1)</td>
</tr>
<tr>
<td>Layer Group-3</td>
<td>(6, 128, 2, 1)</td>
</tr>
<tr>
<td>Layer Group-4</td>
<td>(6, 256, 2, 1)</td>
</tr>
<tr>
<td>FC-1</td>
<td>(16)</td>
</tr>
<tr>
<td>FC-2</td>
<td>(8)</td>
</tr>
<tr>
<td>regressor</td>
<td>-</td>
</tr>
</tbody>
</table>

In the experiments, Adam is used as the optimization algorithm, where the initial learning rate is set as 0.001. The batch size is set as 10. The number of training epoch is 50. Since the model is initialized randomly, all reported experiments are the average of 10 repeats.

2) Pre-processing: The data in the C-MAPSS benchmark is redundant. According to methods [2], [24], the sensor data indexed 1, 5, 6, 10, 16, 18 and 19 is removed. Following the protocol in [2], [24], a sliding window is also applied for data segmentation, where the window length is set as 30 for fair comparison. The piece-wise linear RUL model [2], [24] is also adopted, where the maximal RUL is set as 125. Following Li et al. [7], a min-max normalization is used to normalize the time series data.

3) Experimental Setting: During the experiments, Adam is used as the optimization algorithm, where the initial learning rate is set as 0.001. The batch size is set as 10. The number of training epoch is 50. Since the model is initialized randomly, all reported experiments are the average of 10 repeats.

4) Evaluation: According to methods [2], [24], two metrics, RMSE and scoring function are used to measure the RUL prediction accuracy of our AdaNet. The RMSE can be computed as below:

\[
RMSE = \sqrt{\frac{1}{N} \sum_{i=1}^{N} (\hat{y}_i - y_i)^2}.
\]

For the scoring function, we compute it according to Eq. 10, where more penalization is placed on the late prediction compared to the early prediction.

![Diagram of the turbofan engine](image-url)
### TABLE III
COMPARISON WITH OTHER METHODS.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>FD001</th>
<th>FD002</th>
<th>FD003</th>
<th>FD004</th>
</tr>
</thead>
<tbody>
<tr>
<td>Evaluation</td>
<td>RMSE</td>
<td>Score</td>
<td>RMSE</td>
<td>Score</td>
</tr>
<tr>
<td>Deep LSTM [8]</td>
<td>16.14</td>
<td>338.00</td>
<td>24.49</td>
<td>4450.00</td>
</tr>
<tr>
<td>Huang et al. [24]</td>
<td>N.A.</td>
<td>N.A.</td>
<td>25.11</td>
<td>4793.00</td>
</tr>
<tr>
<td>Li et al. [7]</td>
<td>12.61</td>
<td>273.70</td>
<td>22.36</td>
<td>10412.00</td>
</tr>
<tr>
<td>BLCNN [6]</td>
<td>13.18</td>
<td>302.27</td>
<td>19.09</td>
<td>1558.00</td>
</tr>
<tr>
<td>TCMN [26]</td>
<td>23.57</td>
<td>1220.00</td>
<td>20.45</td>
<td>3100.00</td>
</tr>
<tr>
<td>DGRU [27]</td>
<td>18.54</td>
<td>1467.00</td>
<td>20.06</td>
<td>4085.00</td>
</tr>
<tr>
<td>MODBNE [30]</td>
<td>15.04</td>
<td>334.27</td>
<td>25.05</td>
<td>5585.34</td>
</tr>
<tr>
<td>Jang et al. [28]</td>
<td><strong>12.47</strong></td>
<td>253.00</td>
<td>18.18</td>
<td>1618.00</td>
</tr>
<tr>
<td>AdaNet (ours)</td>
<td>13.12</td>
<td>248.45</td>
<td>15.20</td>
<td>890.71</td>
</tr>
</tbody>
</table>

\[
\text{Score} = \sum_{i=1}^{N} s(i), \quad (10)
\]

where \( s(i) \) is computed as following:

\[
s(i) = \begin{cases} 
1, & \hat{y}_i < y_i \\
\exp\left(\frac{y_i - \hat{y}_i}{\varepsilon} \right) - 1, & \hat{y}_i \geq y_i 
\end{cases}, \quad (11)
\]

### B. Comparison with other methods

In this subsection, we compare the performances of our AdaNet with other state-of-the-art approaches in Table III. The kernel size for the layer group is set as 3 and the feature ratio in \( W_{ce} \) and \( W_{cw} \) is set as 16. The deformable convolution is applied to the layer group-2,1 and layer group-3,2.

Among these approaches listed in Table III, Deep LSTM [8] utilizes LSTM to capture the sequential information, while Huang et al. [24] devises to use the bi-directional LSTM to predict the RUL and shows better performances than Deep LSTM [8]. Compared with these two methods, our AdaNet shows better performances on four subsets. Li et al. [7] develops a CNN to learn the temporal feature to predict the RUL. However, since this method [7] utilizes a fixed architecture of a CNN, it performs much inferior to our AdaNet on these four subsets. This indicates that the fixed CNN cannot well capture the sequential information. In comparison, our AdaNet is able to accurately predict the RUL, benefited from the characteristic of the dynamical network architecture.

BLCNN [6] and TCMN [26] use the hybrid based network, which use both CNN and LSTM to encode the sequential information. Although their network architectures are more complex than our AdaNet which is a CNN based method, their performances are still inferior to ours. In GAN [27], the generative adversarial network is used for data argumentation. In Jang et al. [28], the self-supervised learning is applied for better performances. These auxiliary tasks are used to achieve more accurate prediction on the RUL. Jang et al. [28] shows satisfactory performances on simple subsets FD001 and FD003. However, when it comes to complex scenarios, like data trajectories in FD002 and FD004, our AdaNet performs much better than both Jang et al. [28] and GAN [27]. KNet [2] proposes to use knowledge distillation for better accuracy. It is still surpassed by our AdaNet.

Overall, in these compared approaches, although Jang et al. [28] shows impressive performances on some simple subsets, FD001 and FD003, our proposed AdaNet performs much better than it, when the test scenarios become complex in FD002 and FD004. This verifies our hypothesis that these approaches based on fixed network cannot well adjust to different input time series data. When different condition and fault mode are involved, the performances of these methods are not satisfactory. In comparison, our AdaNet is able to adjust the network architecture, and can well adjust to different conditions and fault modes, achieving the best performances.

### C. Analysis Experiments

In this subsection, we conduct experiments to analyze the performances of our proposed components under different configurations. Among four subsets in the C-MAPSS dataset, FD004 includes six different conditions and two fault modes, which is the most complex subset. We select FD004 to conduct our analysis experiments.

1) Deformable Convolution in Different Layers: The performance of the deformable convolution may depend on the applied layers. In this part, we investigate the performances of the deformable convolution on different layers. The corresponding experimental results are listed in Table IV.

### TABLE IV
ANALYSIS EXPERIMENTS FOR DEFORMABLE CONVOLUTION APPLIED IN DIFFERENT LAYERS ON FD004.

<table>
<thead>
<tr>
<th>Layer</th>
<th>L1</th>
<th>L2,1</th>
<th>L2,2</th>
<th>L3,1</th>
<th>L3,2</th>
<th>L4</th>
<th>RMSE</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>✓</td>
<td>15.64</td>
<td>1017.44</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>✓</td>
<td>15.30</td>
<td>1004.40</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>✓</td>
<td>15.37</td>
<td>973.65</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>✓</td>
<td>16.32</td>
<td>1051.50</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>✓</td>
<td>15.68</td>
<td>975.85</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>✓</td>
<td>15.34</td>
<td>911.64</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>✓</td>
<td>15.54</td>
<td>992.44</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>✓</td>
<td>15.39</td>
<td>932.60</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>✓</td>
<td>15.18</td>
<td><strong>894.35</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>✓</td>
<td>15.81</td>
<td>1029.36</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

To investigate the effectiveness on each layer, we firstly apply the DC layer on a single layer. According to Table IV, a baseline experiment is conducted, which does not use any DC layer and achieves 15.64 RMSE and 1017.44 Score.
After applying a single DC layer to the neural network, nearly all experiments show better performances than the baseline. Among these experiments with a single DC layer, the neural network which applies the DC layer to layer $L_3$, shows the best RUL prediction (911.64) on the Score and the network with a DC layer applied on $L_1$ achieves the best performance (15.30) on the RMSE.

After that, we further investigate the performance of a neural network with more than one DC layer. For experiments with multiple DC layers, the neural network which adopts the DC layer on the layer $L_2$ and layer $L_3$, achieves the best performances on both RMSE (15.18) and Score (894.35). These experiments show that the DC layer can effectively exploit the sequential information and improve the accuracy of the RUL prediction. Since more DC layer involved introduces more computation cost, for computation efficiency, we determine to utilize two DC layers which are applied on the layer $L_2$ and layer $L_3$, separately, in our AdaNet.

2) Deformable Convolution with Different Kernel Size: The performances of deformable convolution may be affected by the kernel size. In this part, we conduct experiments to investigate the influence of the kernel size on the RUL prediction accuracy. For convenience, we use the network which applies the DC layer on the $L_4$, in the following experiments. The experimental results are visualized in Fig. 6.

![Fig. 6. Experiments for deformable convolution with different kernel size on FD004.](image)

From Fig. 6, it can be found that the influence of kernel size on the RMSE and Score are different. The network with $5 \times 5$ kernel achieves the best performances on RMSE, while shows the worst accuracy on the Score. For the network with $3 \times 3$ and $7 \times 7$ kernel size, their performances are similar in both RMSE and Score.

Since the evaluation metric Score places more penalization in the late prediction than the early prediction, the Score metric is more close to the industrial scenario. Compared with RMSE, we prefer to choose a network which achieves satisfactory performances on Score. Among these experiments shown in Fig. 6, in view of Score, the network with $3 \times 3$ kernel size performs similar to the network with $7 \times 7$ kernel size, while performs better than the network with $5 \times 5$ kernel size. For the networks with $3 \times 3$ and $7 \times 7$ kernel size, the network with $3 \times 3$ kernel size requires less parameters and saves computation cost than the network with $7 \times 7$ kernel size. Thus, we believe that the network with $3 \times 3$ kernel size should perform the best among these networks with different kernel size.

Among these experiments, although the number of sampling location in $3 \times 3$ kernel is smaller than that of $7 \times 7$ kernel, their performances on both RMSE and Score are similar. This verifies our hypothesis that the deformable convolution is able to adjust its receptive field according to the input data and shows similar performances to the convolution with larger kernel size.

3) Feature Ratio in Channel Selection Layer: In the CS module, the feature ratio $r$ in $W_{ce}$ and $W_{cw}$ is used for feature dimensionality reduction. To explore its influence on the performance, several experiments are conducted, where a network with a DC layer applied on the $L_4$ is used as a basic method. The experimental results are visualized in Fig. 7.

![Fig. 7. Experiments for channel selection layer with different feature ratio $r$ on FD004.](image)

Compared with the basic method, the RUL prediction is significantly improved on both RMSE and Score after applying the CS module on the network. This indicates that our proposed CD module effectively improves the feature quality and achieves better performance. Moreover, when we increase the feature ratio $r$, the accuracy of the RUL prediction further increases. This indicates that the feature ratio effectively reduces the feature dimension, while improves the feature quality at the same time.

![Fig. 8. Visualization for the RUL predictions on two subsets, FD002 and FD004 of the C-MAPSS benchmark.](image)
D. Visualization of The RUL Prediction Results

In this subsection, we visualize the RUL prediction results on the C-MAPSS benchmark. Among four subsets of the C-MAPSS benchmark, FD002 and FD004 includes different conditions. To show our AdaNet performances on complex scenarios, the predictions from our AdaNet on FD002 and FD004 are visualized in Fig. 8. We can find that the predicted RUL from our AdaNet matches well with the true RUL. This demonstrates that our proposed AdaNet is able to dynamically adjust its architecture according to the input data and accurately predict the RUL.

V. CONCLUSION AND FUTURE WORK

In this paper, we have proposed an adaptive and dynamical neural network (AdaNet) for the RUL prediction. To enable our AdaNet to adjust its architecture online, the deformable convolution and the channel selection module have been developed. The deformable convolution can change the kernel size according to the input data, while the channel selection module enables our AdaNet to adjust the feature channel online. Extensive experiments have been carried out on the C-MAPSS benchmark, which verifies the effectiveness of our proposed AdaNet and shows the state-of-the-art performances achieved by our AdaNet.

The depth of a CNN may also affect the performance of a CNN based approach on the RUL prediction task. In the future, we will investigate how to enable a CNN to adjust its depth according to the input data for better accuracy in the RUL prediction. For example, we can provide multiple paths with different depths in a neural network, like the shortcut connection in ResNet or the inception module in GoogLeNet.

REFERENCES